DSA Concepts Prep:

**Day 1 — Two Sum (Array + HashMap)**

**Difficulty:** Easy  
**Concepts:** Array traversal, complement lookup, hash map

**Problem Recap**

You’re given:

* An array of integers nums
* An integer target

Find **two distinct indices** i and j such that:  
nums[i] + nums[j] == target

Return [i, j].

**Step 1: Understand the Problem**

Example:

nums = [2,7,11,15]

target = 9

We want two numbers that add up to 9 → 2 + 7 = 9  
So, we return their indices [0,1].

**Step 2: Brute-Force Approach (Understand First)**

Try all pairs (i, j) and check if they add up to target.

**Python:**

def twoSum(nums, target):

for i in range(len(nums)):

for j in range(i + 1, len(nums)):

if nums[i] + nums[j] == target:

return [i, j]

Time Complexity: O(n²)  
 Space Complexity: O(1)

This works but is **slow** when n becomes large.

**Step 3: Optimized Approach (Using HashMap)**

Instead of checking all pairs, we use a **hash map** to remember numbers we’ve seen.

**Logic:**

* Loop through each number.
* For each number num, find its **complement** → target - num.
* If complement exists in the hash map → we found our pair.
* Otherwise, store num with its index in the hash map.

**Python Code**

def twoSum(nums, target):

seen = {} # value -> index

for i, num in enumerate(nums):

complement = target - num

if complement in seen:

return [seen[complement], i]

seen[num] = i

**JS Code**

function twoSum(nums, target) {

const seen = new Map();

for (let i = 0; i < nums.length; i++) {

const complement = target - nums[i];

if (seen.has(complement)) {

return [seen.get(complement), i];

}

seen.set(nums[i], i);

}

}

**Step 4: Example Dry Run**

nums = [3, 2, 4], target = 6

| **Step** | **num** | **complement** | **seen map before** | **Found?** | **seen after** |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 3 | 3 | {} | ❌ | {3: 0} |  |  |
| 1 | 2 | 4 | {3: 0} | ❌ | {3: 0, 2: 1} |  |  |
| 2 | 4 | 2 | {3: 0, 2: 1} | ✅ | return [1, 2] |  |  |

**🧮 Complexity Analysis**

| **Type** | **Explanation** |
| --- | --- |
| Time | O(n) — one pass through list |
| Space | O(n) — store elements in hash map |

**Problem:**  
Given nums = [1,5,3,7,9], target = 10  
Return indices of the two numbers that add to 10.

